AADL 端对端数据流一致性验证方法

王 凯[1],毕海滨[2]
(1. 南京航空航天大学 计算机科学与技术学院,江苏 南京 210016;
2. 江南计算技术研究所 第三处,江苏 无锡 214083)

摘 要:复杂嵌入式实时系统的端对端数据流的时延分析是一种有效的实时系统实时性评估方法。体系结构分析与设计语言(Architecture Analysis and Design Language,AADL)是描述实时系统(嵌入式系统)的标准语言,端对端的数据流描述系统组件间的消息传递。提出一种基于 Prolog 的端对端数据流分析方法,解决嵌入式实时系统的 AADL 模型时延验证问题。针对 AADL 模型缺乏时延验证的现状,分析讨论了 AADL 模型的端对端数据流并提出了端对端数据流的路径一致性的定义;针对单一型端对端数据流和混合型端对端数据流给出了两种端对端数据流到基本状态图的映射方法;设计了端对端数据流路径一致性的 Prolog 验证规则。最后对带时间约束的汽艇速度控制子系统进行实例验证,结果表明该方法能够有效地解决实时系统的时延验证问题。

关键词:端对端数据流;实时系统;一致性;Prolog
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2018)01-0001-05
doi:10.3969/j.issn.1673-629X.2018.01.001

A Consistency Validation Approach of AADL End-to-end Flow Latency

WANG Kai[1],BI Hai-bin[2]
(1. School of Computer Science and Technology,Nanjing University of Aeronautics & Astronautics,
Nanjing 210016,China;
2. Third Department of Jiangnan Institute of Computing Technology,Wuxi 214083,China)

Abstract:Time-delay analysis of end-to-end data streams of complex embedded real-time systems is an effective method of real-time system evaluation. The Architecture Analysis and Design Language (AADL) is a standard language that describes real-time systems (embedded systems),and end-to-end data streams describe message passing between system components. An end-to-end data flow analysis method based on Prolog is proposed to solve the problem of AADL model latency verification for embedded real-time system. In view of the lack of time delay verification of AADL model,the end-to-end data flow of AADL model is analyzed and discussed,and the definition of path consistency of end-to-end data flow is proposed. For single end-to-end and hybrid end-to-end data flow,two mapping methods of end-to-end data flow to basic state diagram are given. A Prolog authentication rule is designed for end-to-end data flow path consistency. Finally,the time control of the speed control subsystem with time constraints is verified. The results show that the method can effectively solve the problem of real-time system delay verification.

Key words:end to end data flow;real-time system;consistency;Prolog

0 引 言

在针对实时系统[1]的 OSATE[2]建模过程中,数据和事件的传输通过数据流完成。一条完整的数据流通常由一个数据采集器开始,流经中间组件(进程组件),到控制器结束。数据传输过程具有一定的时效性,系统会因为数据流不能及时到达而失效。

AADL 标准的制定者 PETER 提出影响数据流时延的因素主要有四点[3]:组件本身的计算时延;不同组件间的传输时延;数据采样速率和设备端口上数据队列的处理方式;传输协议对于数据等待队列的处理方式。作者在理论层面上全面分析了时延的影响因素,但只是选取了 AADL 属性集合中的 Latency 属性作为分析的依据,对端对端的数据流做了简单的模拟。LEE Su-Young[4]基于进程组件中的周期和非周期线

收稿日期:2016-12-30 **修回日期**:2017-04-27 **网络出版时间**:2017-09-27
基金项目:"十三五"重点基础科研项目(JCKY2016206B001)
作者简介:王 凯(1984-),男,硕士研究生,研究方向为软件体系结构。

程给出了数据流的最优时延与最差时延的计算公式，但对于系统的调度并没有给出分析依据，例如多线程的优先级。谯婷婷等[5]通过实例给出了数据流分析的结果，但没有给出一个通用的解决方案。文献[6-7]将 AADL 模型转换成时间自动机等模型来进行性质的等价验证，用 CPS 进行描述，用 PDA 工具进行流性质验证，不过少有提及调度对端对端数据流的影响。

文中以数据流时延特性为研究目标，建立数据流到 Prolog 的转换，对于流经线程的数据流，结合调度模型对多线程优先级的考虑，针对单一型数据流及混合型数据流验证了实时系统端对端数据流的一致性。

1　AADL 端对端数据流及一致性定义

端对端的数据流[8]（end to end flow）描述系统内部数据和事件的传递路径。AADL 中完整的端对端数据流定义应包含两个部分：流声明（flow specification）以及流实现（flow implementation）。

1.1　端对端数据流声明

流声明依附在组件的声明中完成，定义从组件的输入到输出的逻辑路径。流的基本要素为 flow source（流源）、flow sink（流汇）和 flow path（流路径）。其中 flow source 和 flow sink 是组件 features 定义下的端口（ports）、端口组（ports group）或者共享参数（parameter），flow path 是 flow source 和 flow sink 间的连接。

1.2　端对端数据流实现

与流的声明一样，流的实现依附在组件的实现（implementation）中，通过组件与子组件、子组件与子组件的串联，形成了从输入端口到达输出端口的串行序列，确切描述了组件内部数据的流向。AADL 定义规定，可以建立数据端口（data port）到数据、事件端口（event port）或数据事件端口（event data port）的连接，而事件端口仅可以和事件端口单向连接，事件端口向数据或者数据事件端口的连接是不允许的，例如流路径能够从任意类型的流入端口流向不同类型的端口（从数据端口流向事件端口）。

一个端对端的数据流 Fp 的定义可以描述为一个六元组，$Fp=\{F, f_s, f_d, Po, Co, La\}$。其中，$F$ 为作用于组件内部的数据流，其表达式为 $F=\{Po \times Po, La\}$，端对端的数据流也可表述为 $Fp=\{F \times F, f_s, f_d\}$；$f_s \in F$，表示 flow source，数据流的起点；$f_d \in F$，表示 flow sink，是数据流的终点；Po 为端口组件的集合，$Po=\{$data port，event port，eventdata port$\}$，代表数据端口、事件端口和事件数据端口；$Co=\{Po \times Po, \varphi'\}$，表示不同组件间的端口连接的集合，$\varphi'$ 为连接约束的条件，禁止 event port 指向 data port 或 event port，而其他指向的连接都是合法的；La 表示 Latency 属性的集合，它作用于 F 和 Co。

1.3　数据流时延相关属性

在不考虑组件本身是线程的情况下，计算整个端对端数据流时延时，与其直接相关的因素是 Latency 属性[9-10]。Latency 属性代表数据流在端口间最长的时间消耗。在 AADL 的属性集合中，定义了与数据流时延直接相关的属性 Latency，用于规定在流或连接上所允许的时间延迟，描述的对象包括数据流（flows）、连接（connections）等；属性赋值为时间类型（整形加上时间单位）标准中给出的解释。

1.4　端对端数据流的路径一致性

定义 1：端对端数据流的路径一致性是指在一条完整的端对端数据流 Fp 中，针对数据流路径（flow path）上的所有端口组件 Po（普通端口组件 Po 满足对应的 Compute_Execute_Time 属性）及端口间的数据流（数据流满足对应的 Latency 属性），任意一组时刻均满足该数据流路径上的所有约束。

图 1 为汽艇自动驾驶仪系统的一部分。其中，speed_sensor 是速度传感器，数据从 sensor_data 端口经过数据流到速度计算进程 speed_control 的 sensor_data 端口，通过组件内的数据流到达 speed_control 的 command_data 端口，最后经数据流到达控制设备 throttle 的 command_data 端口，总共包含了四个数据端口，两个组件间的数据流以及一个组件内部的数据流。图中，speed_sensor 数据采集设备中的数据端口 sensor.data 通过数据流进入下一个数据端口 speed_control 进程中的 sensor_data 数据端口，这两个组件间的数据流规定其 Latency 属性为[5 ms，5 ms]，表示数据流时延在此时间范围内。而对于整个端口到端口的数据流，系统规定的时间时延（constraint）为 35 ms，即对应的从设备 speed_sensor 的 sensor_data 数据端口到设备 throttle 的 command_data 数据端口所消耗的时间应在 35 ms 内。如果存在一组时间取值满足端对端的数据流在 35 ms 内完成，则此一致性是满足的，否则说明此端对端数据流路径不一致（这里不考虑组件本身的 Compute_Execute_Time，仅针对数据流时延讨论）。

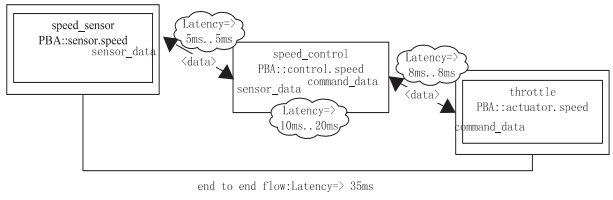

图 1　PBA 子系统的数据计算及转发

2　基于 Prolog 的单一型 & 混合型端对端数据流验证方法

作为一种描述型语言，Prolog[11] 只需描述问题中

的对象和对象关系等已知的事实与规则,确定对象间的逻辑关系。给出待验证的实时系统的事实与规则,作为 Prolog 的数据(即程序,数据与程序是统一的)进行性质验证。为清楚地表述端对端数据流的 Prolog 事实,先将端对端数据流映射成状态图[12],整条数据流映射为一个从初始化状态到结束状态的闭环。

根据端对端数据流所在的层级、是否流经进程下面具体的线程和调度程序是否对线程进行派发,介绍基于 Prolog 的单一型端对端数据流及混合型端对端数据流[13](以下简称单一型数据流和混合型数据流)的验证方法。

2.1　单一型数据流映射方法

单一型数据流是指在端对端数据流中,数据流经组件时不考虑流向其子组件且流经进程组件时,不考虑线程任务本身的计算时延和调度器对线程调度的影响,设备对数据以及事件信号的发送和接收是即时的,数据流时延的产生仅考虑流属性中定义的 Latency。例如,图 1 中的传感器 speed_sensor 将采集的速度信息通过数据端口 sensor_data 发送给处理器 speed_control,处理器通过 sensor_data 数据端口接收来自传感器的速度信息,计算结束后通过数据端口 command_data 将速度信息传递给执行器设备 throttle 的数据端口 command_data,由速度执行器对汽艇速度做出调整。在 PBA 子系统的端对端数据流中包含了四个数据端口,对应这两个组件间的数据流及一个组件内部的数据流。对于此数据流,基于 Prolog 进行验证的规则如下所述:

(1)Po 数据端口映射为基本状态,按照实际系统中数据的流向串联起来,在 flow source 前加入 Start 初始化状态表述整条数据流的开始,且在 flow sink 后加入中止状态 End 作为验证数据流路径一致性的参考状态。

(2)La 表示数据流的 Latency 属性,映射为数据端口间数据流时延信息,设置相应的整形变量表示其时间约束。

(3)Co 表示 Po 组件间的连接,映射到基本状态信息中各状态间的连接。

(4) *F* 表示组件内部的数据流,例如进程中各个线程间的数据流交互。在单一型数据流不考虑调度策略的情况下,转换为当前组件的两端口间的数据流,在基本状态图中表示为状态间的连接。

PBA 系统数据采集及转发的单一型数据流映射为基本状态信息如图 2 所示。其中,PBA_Sensor_Data、PBA_Control_Data、PBA_Control_cmdData 及 PBA_Throttle_cmdData 分别表示 PBA 子系统中的四个数据端口。在 flow source 前加入 PBA_SpeedControl_Start 初始化状态及在 flow end 后加入 PBA_SpeedControl_End 中止状态。基本状态间的数据流包含 Latency 时延信息。

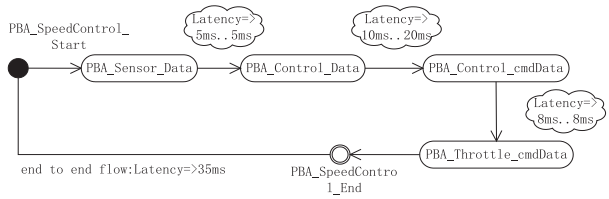

图 2　单一型数据流的基本状态图

2.2　带线程优先级的混合型数据流映射方法

由数据采集器产生、流经线程或者触发非周期事件派发的数据流为混合型数据流。文中讨论流经线程的数据流的时延情况,接着介绍在基于线程优先级调度策略下混合型数据流的验证方法。

对 PBA 控制子系统在图 2 中做了部分扩充,如图 3 所示。

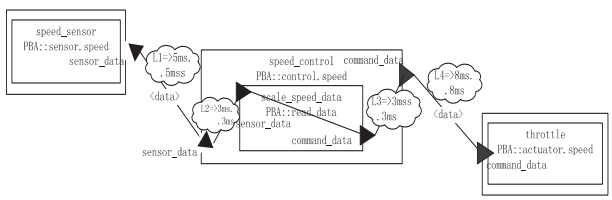

图 3　包含计算线程的 PBA 子系统

数据流经过进程 speed_control 中的计算线程 scale_speed_data,线程计算后将结果返回给进程 command_data 接口,数据流时延的计算必须考虑计算线程的影响,而线程从调度系统派发到执行结束受调度状态的影响。

传感器 speed_sensor 任务派发执行后产生数据,经过 speed_sensor 数据端口到达计算进程的 speed_control 的数据端口 sensor_data,并从 sensor_data 数据端口传送到计算线程 scale_speed_data 中的 sensor_data 数据端口,对应的两条数据流的 Latency 属性分别记为 L_1 和 L_2。在到达计算线程 scale_speed_data 时,触发了线程的派发,由于该线程的优先级不高,得不到计算。图 4 表示 PBA 控制子系统的数据流时延情况。虚线部分代表计算线程因优先级不高而得不到计算的等待时间。随后数据经过 Thread_Execution_time 的计算时间后经过数据流 L_3、L_4 到达执行器 throttle。

基于状态图的混合型数据流映射规则如下:

(1)Po 数据端口映射为基本状态,按照实际系统中数据的流向串联起来,并在 flow source 前加入 Start 初始化状态表述整条数据流的开始,并在 flow sink 后加入中止状态 End 作为验证数据流路径一致性的参考状态。

(2)La 表示数据流的 Latency 属性,映射为数据端口间的数据流时延信息,设置相应的整形变量表示其

时间约束。

(3)Co 表示 Po 组件间的连接,映射到基本状态信息中各状态间的连接。

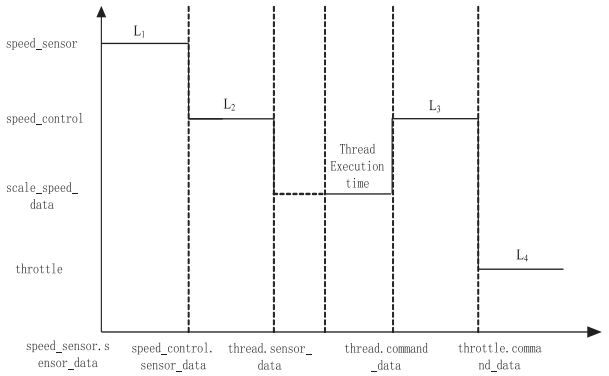

图 4　PBA 子系统时延分析示意图

(4) F 表示组件内部的数据流,带线程优先级的混合型数据流中,按照线程优先级的先后顺序将优先级高的线程(包含待计算的线程)映射为基本状态加入组件内部数据流中,由线程映射的基本状态本身带有线程计算所产生的时延 Thread_Execution_time。

图 5 为 PBA 子系统混合型数据流的基本状态图,其中 PBA_ScaleThread 线程为最高优先级线程,映射为基本状态且线程的执行时间为 8 μs。

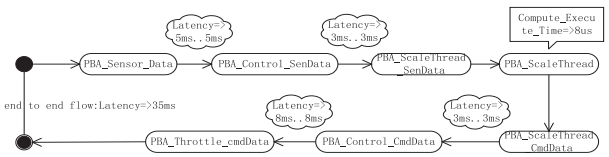

图 5　混合型数据流的基本状态图

2.3　基于 Prolog 的数据流一致性验证规则

针对端对端数据流 Prolog 验证规则,利用 minbordercons 函数及 maxbordercons 函数描述 1.4 节中路径一致性问题,下面给出函数具体实现及验证表达。

使用 minbordercons(A,B,T) 函数来计算在给定的相邻的基本状态(即 Po 端口) A,B 通信时所要花费的最短时间,并将计算后的结果赋值给 T;minroutecons(A,B,T) 函数则通过递归的思想计算端对端数据流路径中组件之间的最短时延;fact(A,B,S,E) 表示基本状态图中状态节点的 Prolog 事实表达,A,B 表示基本状态,S,E 表示基本状态 A,B 间数据流时延的最小值及最大值;contraint(A,B,S) 表示基本状态 A,B 间的时间约束。

Prolog 程序如下:

```
minbordercons( A,B,T ):-not( A = B ),
fact( A,B,S1,_ ),
fact( B,_,S2,_ ),
T is S1 + S2.
minroutecons( A,B,T ):-
fact( A,Z,S1,_ ),
minroutecons( Z,B,T1 ),
T is S1 + T1.
```

使用 maxbordercons(A,B,T) 来计算在给定的相邻的基本状态 A,B 所需要花费的最长时间,并将计算后的结果赋值给 T;maxroutecons(A,B,T) 函数则通过递归的思想计算端对端数据流路径中组件之间最长时延,Prolog 程序如下:

```
maxbordercons( A,B,T ):-
not( A = B ),fact( A,B,_,E1 ),
fact( B,_,_,E2 ),
T is E1 + E2.
maxroutecons( A,B,T ):-
not( A = B ),fact( A,B,_,E1 ),
fact( B,_,_,E2 ),
T is E1 + E2.
maxroutecons( A,B,T ):-
fact( A,Z,_,E1 ),
maxroutecons( Z,B,_,T1 ),
T is E1 + T1.
```

结合上述函数,给出端到端数据流的路径一致性验证的 Prolog 表达。

```
fact(pba_start,sensor_data,0,0)
```

3　实例验证与分析

本节将以汽艇自动驾驶仪(PBA)系统中的速度控制模块为例,基于 Prolog 对 AADL 端对端的数据流进行一致性验证。

3.1　实例验证

建立一个速度控制的 AADL 模型[14],其功能是通过传感器采集数据,通过数据端口发送给进程的控制端口用来计算当前的最佳艇速,最后通过数据端口传输给执行器,用来改变当前艇速。

针对速度控制子系统的单一型数据流及混合型数据流根据 2.1 及 2.2 提出的转换方法,转换后的基本状态图见图 4 及图 5。下面给出单一型数据流及混合型数据流基本状态图的 Prolog 事实表达。

```
fact(pba_start,sensor_data,0,0)
fact(sensor_data,control_data,5,5)
fact(control_data,control_cmddata,10,20)
fact(control_cmddata,throttle_cmddata,8,8)
fact(throttle_cmddata,pba_end,0,0)
constraint(pba_start,pba_end,35)
fact(pba_start,sensor_data,0,0)
fact(sensor_data,control_sendata,5,5)
```

```
fact(control_sendata,scaleThread_sendata,3,3)
fact(scaleThread_sendata,scaleThread,8,8)
fact(scaleThread,scaleThread_cmddata,0,0)
fact(scaleThread_cmddata,control_cmddata,3,3)
fact(control_cmddata,throttle_cmddata,8,8)
fact(throttle_cmddata,pba_end,0,0)
constraint(pba_start,pba_end,35)
```

将上述事实与第 2 节所提的路径一致性验证规则作为 Prolog 程序的输入,得到的实验数据如表 1 所示。

表 1　数据流路径一致性实验结果

数据流	流源	流汇	时延约束/ms	结果/ms	满足一致性
单一型	start	end	35	[23,33]	满足
混合型	start	end	35	[27,27]	满足

从表 1 中可以看到,PBA 速度控制子系统中的单一型及混合型数据流完成数据传输需要消耗的时间分别为[23 ms,33 ms]和[27 ms,27 ms],相对于整个系统的时延约束 35 ms,数据流传输满足了系统的要求。

3.2　实例分析

针对现有的实时系统时间一致性验证的不足,提出了基于 Prolog 的 AADL 端对端数据流路径一致性验证方法。对端对端数据流进行分类描述,给出端对端数据流路径一致性 Prolog 验证表达,最后通过 SWI-Prolog 进行验证。该方法和其他类似建模验证方法在验证方面的比较如表 2 所示,其中 yes 表示支持该性质,no 表示不支持。

表 2　与现有方法的对比

方法	单一型数据流路径一致性	混合型数据流路径一致性	基于线程优先级的调度转换
PETER[3]	yes	no	no
LEE Su-Young[4]	yes	yes	no
谯婷婷[5]	yes	yes	no
ZHUYufeng[6]	yes	no	no
文中方法	yes	yes	yes

从表 2 可以看出,针对现有的方法去验证实时系统的时延特性时,现有方法很少能将其统一表述在同种方法中进行验证。

4　结束语

文中提出一种基于 Prolog 的端对端数据流分析方法。首先提出实时系统时延特性的路径一致性问题,其次针对单一型数据流及混合型数据流特点设计了基于 Prolog 的验证方法,最后通过将模型转换成 Prolog 事实及路径一致性转换成 Prolog 规则,实现了对实时系统时延性质的验证。

参考文献:

[1] HAI N T,SINGHOFF F,RUBINI S,et al. Instruction cache in hard real-time systems:modeling and integration in scheduling analysis tools with AADL[C]//International conference on embedded and ubiquitous computing. [s. l.]:IEEE,2014:104-111.

[2] LOUKIL S,KALLEL S,ZALILA B,et al. AO4AADL:aspect oriented extension for AADL[J]. Central European Journal of Computer Science,2013,3(2):43-68.

[3] ZHAO Y,MA D. Embedded real-time system modeling and analysis using AADL[C]//International conference on networking and information technology. [s. l.]:IEEE,2010:247-251.

[4] LEE S Y,MALLET F,SIMONE R D. Dealing with AADL end-to-end flow latency with UML MARTE[C]//International conference on engineering of complex computer systems. [s. l.]:IEEE Computer Society,2008:228-233.

[5] 谯婷婷,王　乐,耶国栋. 基于 AADL 的软件可靠性验证[J]. 计算机应用,2012,32(S2):92-95.

[6] ZHU Y,DONG Y,MA C,et al. A methodology of model-based testing for AADL flow latency in CPS[C]//5th international conference on secure software integration & reliability improvement companion. [s. l.]:IEEE,2011:99-105.

[7] YU H,YANG Y. Latency analysis of automobile ABS based on AADL[C]//International conference on industrial control and electronics engineering. [s. l.]:[s. n.],2012:1835-1838.

[8] SKELIN M,GEILEN M,CATTHOOR F,et al. Worst-case latency analysis of SDF-based parametrized dataflow MoCs[C]//Proceedings of design and architectures for signal and image processing. [s. l.]:IEEE,2015:1-6.

[9] MAGNON A. Time-latency,time-flow and universal blueprint[M]. [s. l.]:[s. n.],2015:197-217.

[10] 朱晨曦,张立臣,罗崇伟. 基于 AADL 的 CPS 系统分析与设计[J]. 计算机应用与软件,2015,32(8):94-98.

[11] 李　娜,王湘云. 基于谓词逻辑的 Prolog 程序设计[J]. 西南大学学报:社会科学版,2009,35(6):48-52.

[12] 樊　波,袁国铭,周　萍,等. UML 状态图在软件工程设计中的应用研究[J]. 微型电脑应用,2015,31(11):36-37.

[13] 白海洋. 基于 UPPAAL 的嵌入式系统 AADL 模型实时性验证[D]. 南京:南京航空航天大学,2014.

[14] 余晃晶,李仁发,黄丽达. 基于 AADL 的汽车防滑控制系统可调度性分析[J]. 湖南大学学报:自然科学版,2012,39(3):43-47.